A Scanner for a Small JavaScript

ITP40004 – 01 Compiler Theory

21600193 Hyorim, Kim

Apr 10th, 2020

1. regular expression

1) Digit

(1) [0-9]

2) Letter

(2) ‘_’ ([a-zA-Z]| [0-9] |‘_’ | ‘$’)+ | ([a-zA-Z]| ‘$’)* ([a-zA-Z]| [0-9] | ‘_’ | ‘$’)+

3) Identifier

(3) identifier = letter(letter | digit | ‘.’)*

4) Reserved Word and Identifier

(1) reserved = while | if | for | do | switch | case | then | else | break | out.println | main | class

(2) identifier = int

5) Comment

(1) [^//][a-zA-Z0-1]*[(~newline)$]+

- begin with a ‘//’ characters and continue to the end of the line

6) White Space

(1) whitespace = (newline | blank | tab | comment)

2. NFA

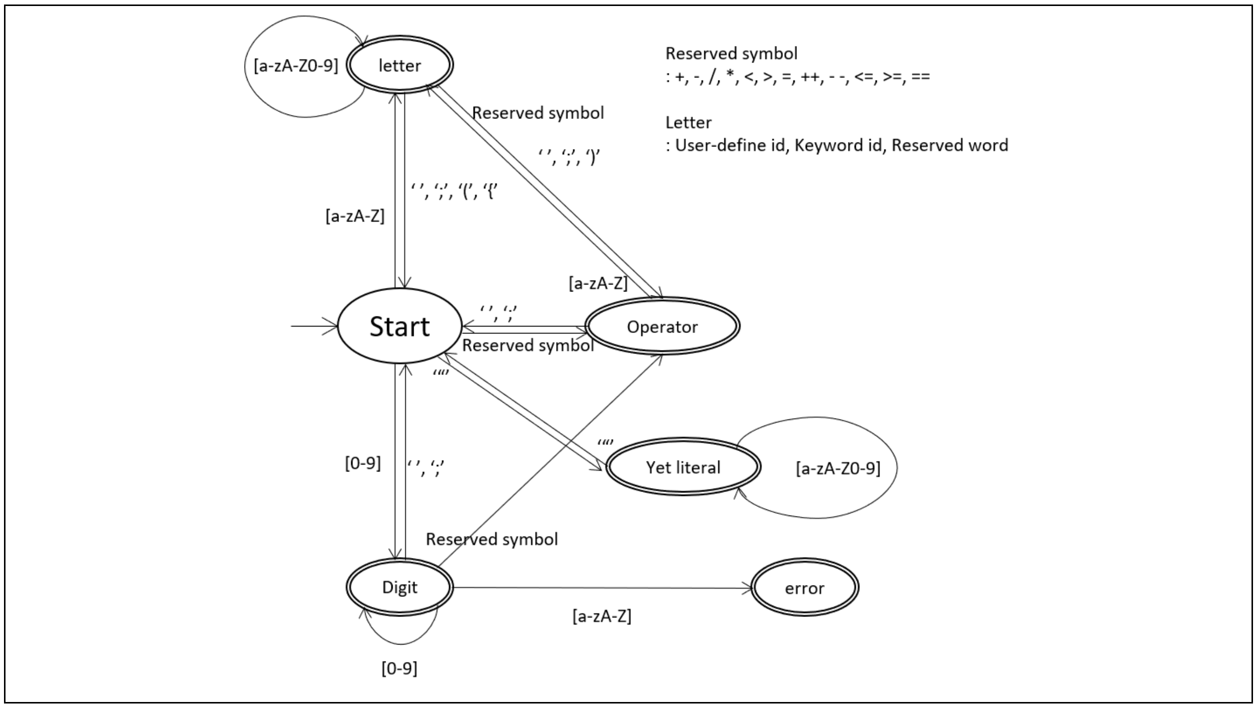


Figure 1: NFA notation for scanner

3. DFA

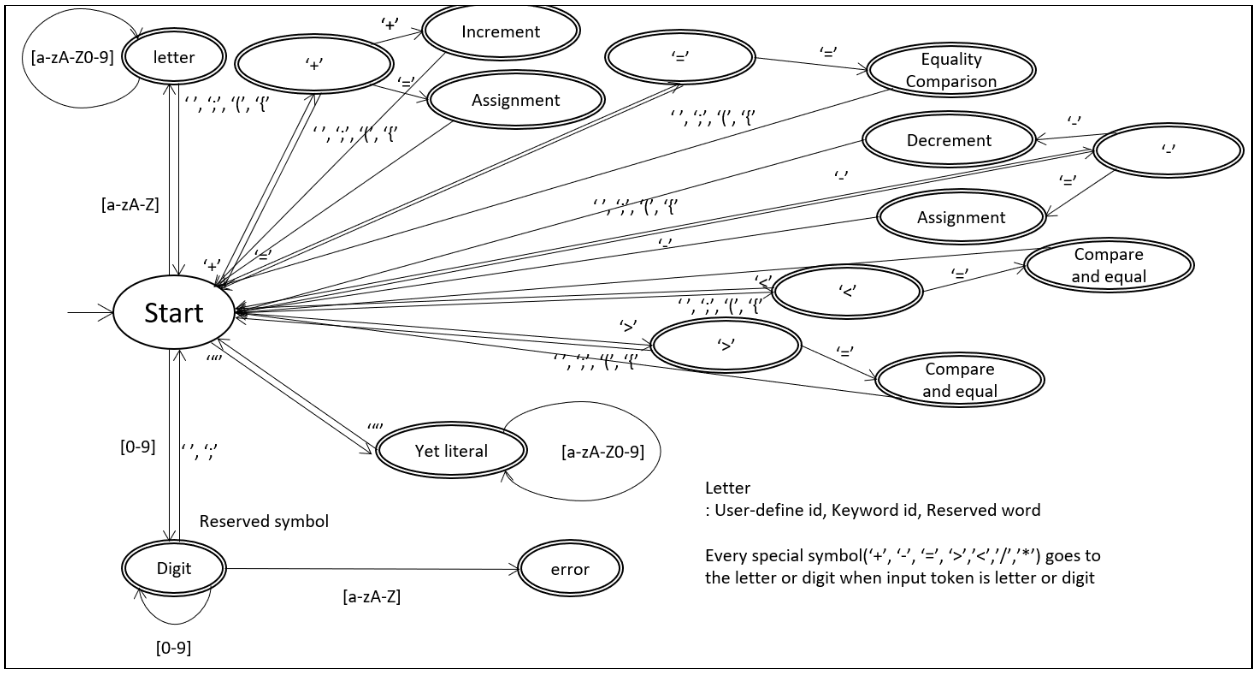


Figure 2: DFA notation for scanner

4. JAVA Source Code

5. Design Document

1) UML

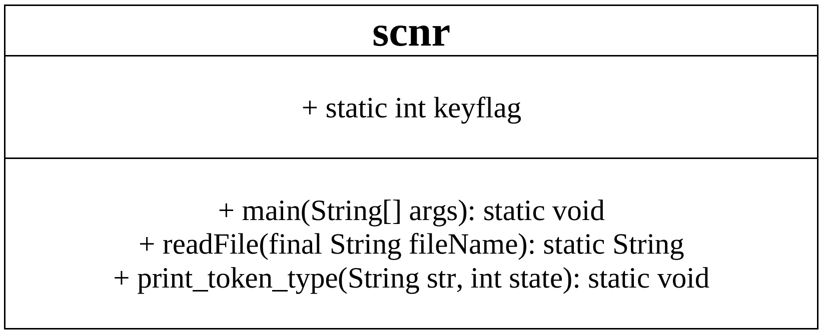

scnr
+ static int keyflag
+ main(String[] args): static void
+ readFile(final String fileName): static String
+ print_token_type(String str, int state): static void

Figure 3: UML of scnr java code

- The scnr.java code has a figure 2 UML.

- public static int keyflag: Stores the keyword properties of a token

- public static void main function: Acts as the main driver. Distinguish type of each tokens.

- public static String readFile function: Reads the file and converts it to a static String.

- public static void print token_type: Print each token with the type.

2) Flow

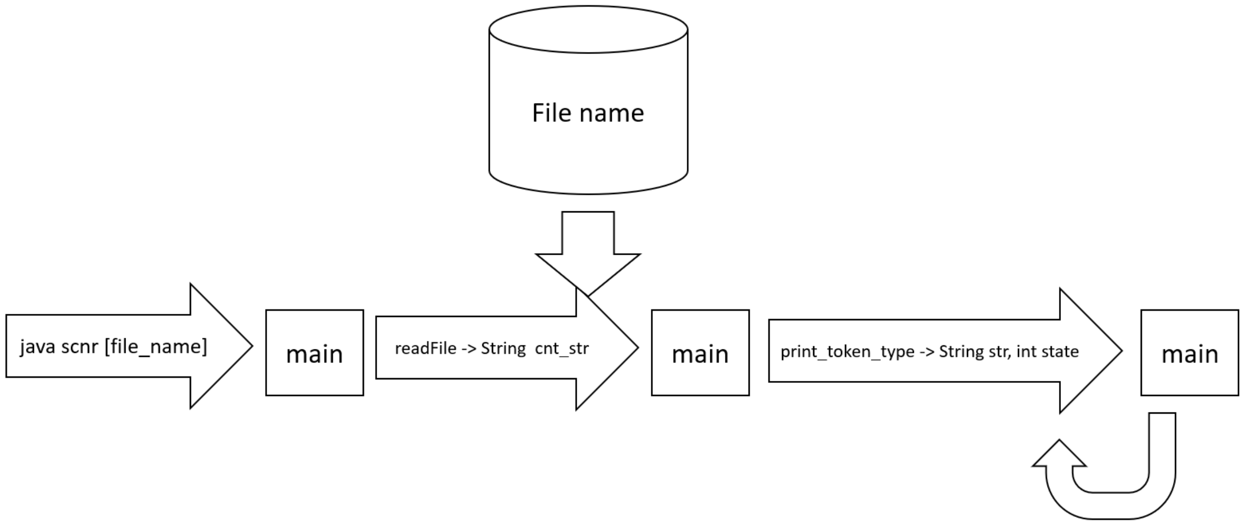

File name
java scnr [file_name]
main
readFile -> String cnt_str
main
print_token_type -> String str, int state
main

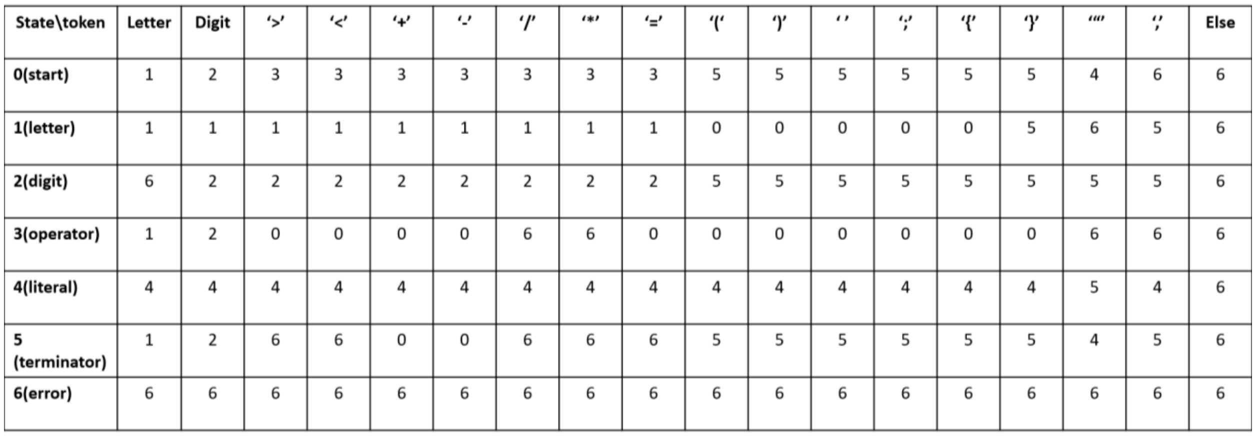
Figure 4: Flow of scnr java program

- When scnr is executed, it goes to the main function first.

- The main function takes the name of the file to be scanned as an argument.

- Put the file name in the readFile function, convert the contents of the file to String, and return to main to scan the String.The pseudo-code of the algorithm used for scanning is the same as in 3 with Figure 5).

- When the token is separated with the type, it outputs with print_token_type function and returns to main. Repeat until there are no remaining characters in buffer.



State\token	Letter	Digit	‘>’	‘<’	‘+’	‘-’	‘/’	‘*’	‘=’	‘(‘	‘)’	‘ ’	‘;’	‘{’	‘}’	‘“”’	‘,’	Else
0(start)	1	2	3	3	3	3	3	3	3	5	5	5	5	5	5	4	6	6
1(letter)	1	1	1	1	1	1	1	1	1	0	0	0	0	0	5	6	5	6
2(digit)	6	2	2	2	2	2	2	2	2	5	5	5	5	5	5	5	5	6
3(operator)	1	2	0	0	0	0	6	6	0	0	0	0	0	0	0	6	6	6
4(literal)	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	6
5 (terminator)	1	2	6	6	0	0	6	6	6	5	5	5	5	5	5	4	5	6
6(error)	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6

Figure 5: Transition Table

- A table representing the transition state corresponding to the DFA in 3.

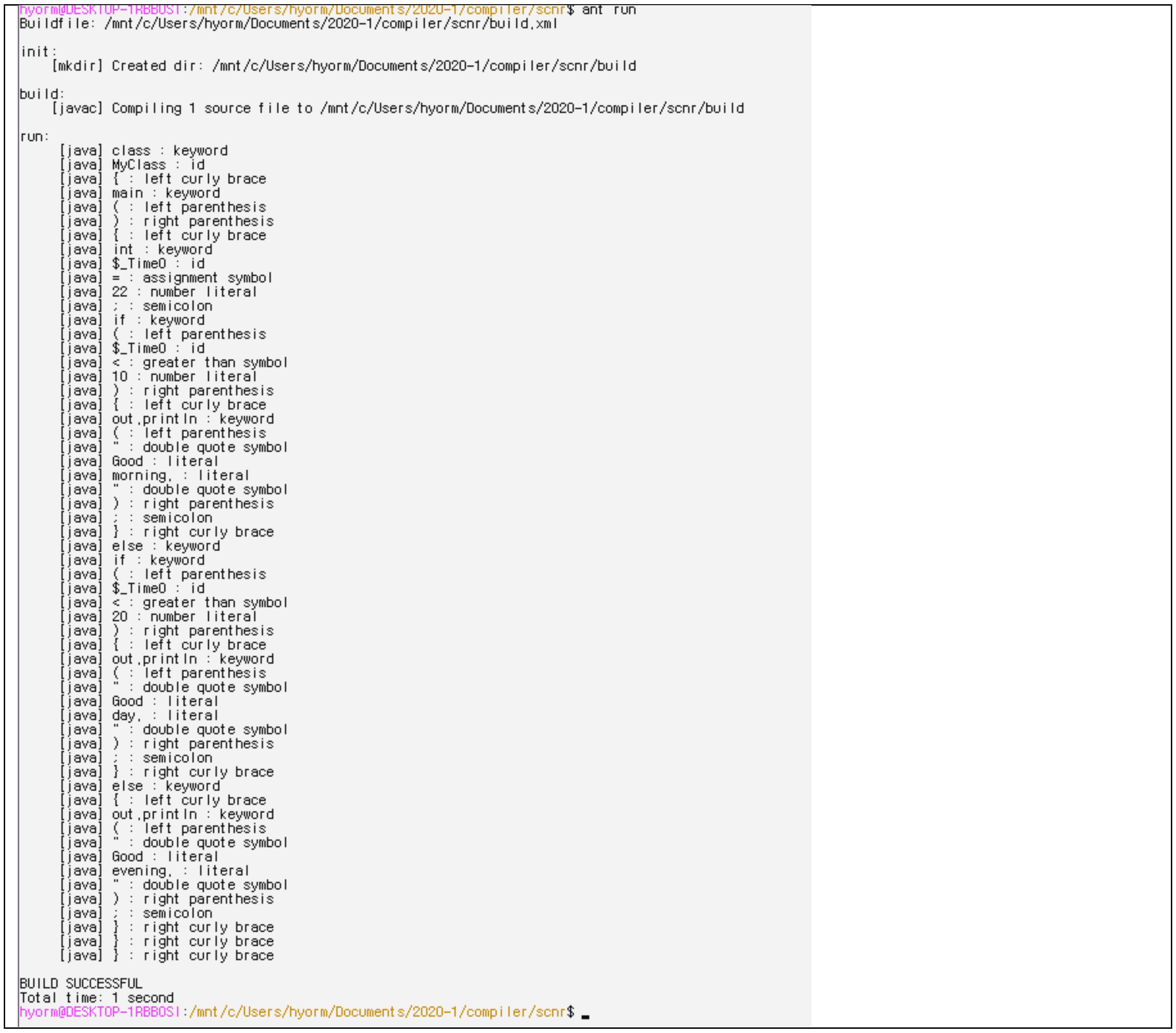
- This table does not show an acceptance state.

3) Pseudocode

```
main
state:= 0
terminator = '(', ')', ' ', ';', '{', '}', '"', ','
special_symbol = '+', '-', '=', '<', '>', '/', '*'
ch := next input character;
while ch is not empty do
    if ch is digit
        state = trs_tbl[state+1][digit];
    else if ch is letter
        state = trs_tbl[state+1][letter];
    else if ch is terminator
        print_token_type();
        state = trs_tbl[state+1][terminator];
    else if ch is special_symbol
        print_token_type();
        state = trs_tbl[state+1][special_symbol];
    else
        error occur;
end while;
```

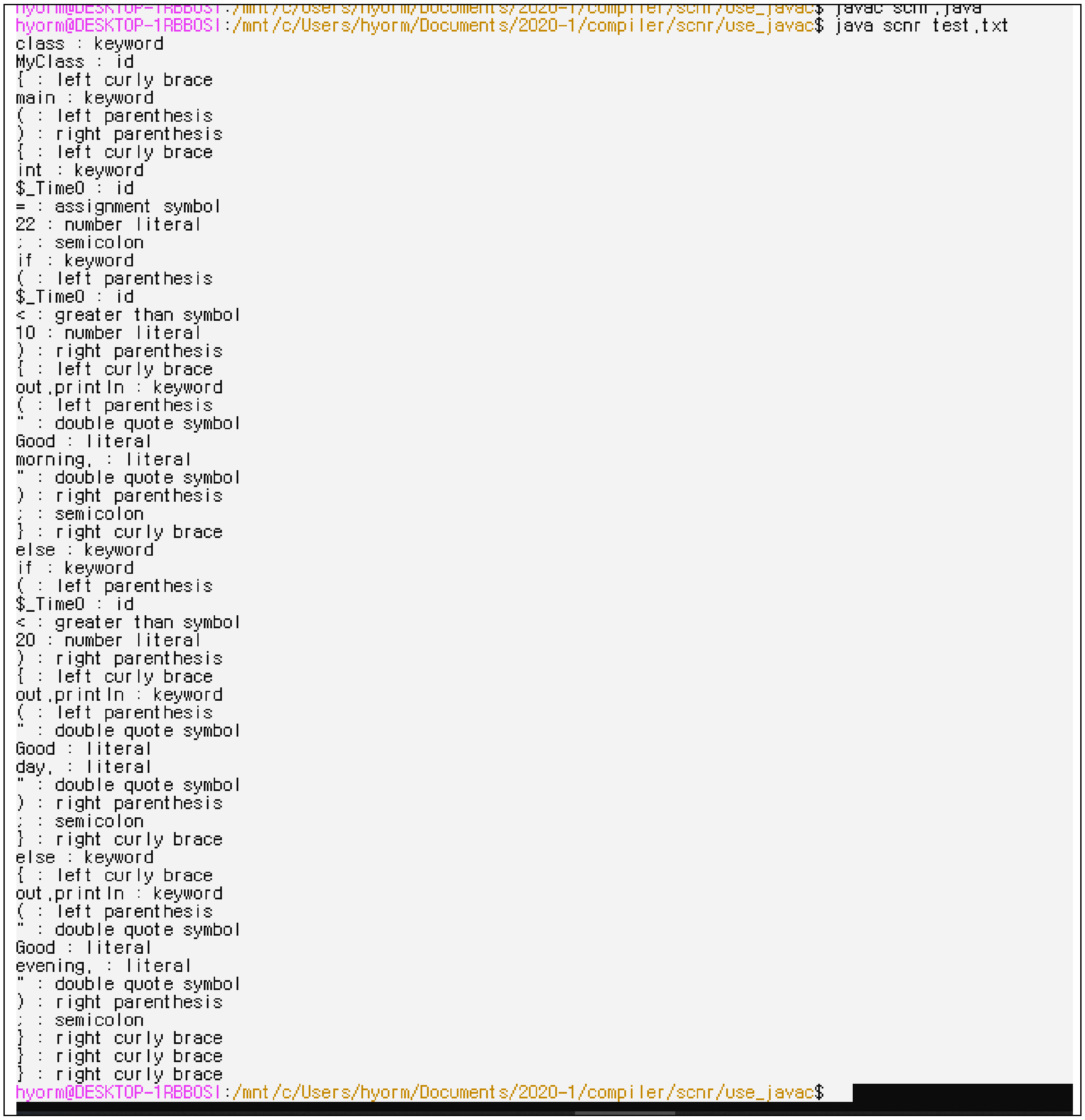
6. Snapshot of Running Result

1) Using ant



```
hyorm@DESKTOP-1RBBOSI:/mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr$ ant run
Buildfile: /mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/build.xml

init:
    [mkdir] Created dir: /mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/build

build:
    [javac] Compiling 1 source file to /mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/build

run:
     [java] class : keyword
     [java] MyClass : id
     [java] { : left curly brace
     [java] main : keyword
     [java] ( : left parenthesis
     [java] ) : right parenthesis
     [java] { : left curly brace
     [java] int : keyword
     [java] $_Time0 : id
     [java] = : assignment symbol
     [java] 22 : number literal
     [java] ; : semicolon
     [java] if : keyword
     [java] ( : left parenthesis
     [java] $_Time0 : id
     [java] < : greater than symbol
     [java] 10 : number literal
     [java] ) : right parenthesis
     [java] { : left curly brace
     [java] out.println : keyword
     [java] ( : left parenthesis
     [java] " : double quote symbol
     [java] Good : literal
     [java] morning, : literal
     [java] " : double quote symbol
     [java] ) : right parenthesis
     [java] ; : semicolon
     [java] } : right curly brace
     [java] else : keyword
     [java] if : keyword
     [java] ( : left parenthesis
     [java] $_Time0 : id
     [java] < : greater than symbol
     [java] 20 : number literal
     [java] ) : right parenthesis
     [java] { : left curly brace
     [java] out.println : keyword
     [java] ( : left parenthesis
     [java] " : double quote symbol
     [java] Good : literal
     [java] day, : literal
     [java] " : double quote symbol
     [java] ) : right parenthesis
     [java] ; : semicolon
     [java] } : right curly brace
     [java] else : keyword
     [java] { : left curly brace
     [java] out.println : keyword
     [java] ( : left parenthesis
     [java] " : double quote symbol
     [java] Good : literal
     [java] evening, : literal
     [java] " : double quote symbol
     [java] ) : right parenthesis
     [java] ; : semicolon
     [java] } : right curly brace
     [java] } : right curly brace
     [java] } : right curly brace

BUILD SUCCESSFUL
Total time: 1 second
hyorm@DESKTOP-1RBBOSI:/mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr$
```

2) Using javac



```
hyorm@DESKTOP-1RBBOSI:/mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/use_javac$ javac scnr.java
hyorm@DESKTOP-1RBBOSI:/mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/use_javac$ java scnr test.txt
class : keyword
MyClass : id
{ : left curly brace
main : keyword
( : left parenthesis
) : right parenthesis
{ : left curly brace
int : keyword
$_Time0 : id
= : assignment symbol
22 : number literal
; : semicolon
if : keyword
( : left parenthesis
$_Time0 : id
< : greater than symbol
10 : number literal
) : right parenthesis
{ : left curly brace
out.println : keyword
( : left parenthesis
" : double quote symbol
Good : literal
morning, : literal
" : double quote symbol
) : right parenthesis
; : semicolon
} : right curly brace
else : keyword
if : keyword
( : left parenthesis
$_Time0 : id
< : greater than symbol
20 : number literal
) : right parenthesis
{ : left curly brace
out.println : keyword
( : left parenthesis
" : double quote symbol
Good : literal
day, : literal
" : double quote symbol
) : right parenthesis
; : semicolon
} : right curly brace
else : keyword
{ : left curly brace
out.println : keyword
( : left parenthesis
" : double quote symbol
Good : literal
evening, : literal
" : double quote symbol
) : right parenthesis
; : semicolon
} : right curly brace
} : right curly brace
} : right curly brace
hyorm@DESKTOP-1RBBOSI:/mnt/c/Users/hyorm/Documents/2020-1/compiler/scnr/use_javac$
```

7. User Manual

1) Use ant (directory name)

(1) ant version

- Apache Ant(TM) version 1.10.5 compiled on March 28 2019

(2) build.xml

```
<project name="scnr" default="build" basedir=".">

    <property name="src" value="src"/>
    <property name="build" value="build"/>
    <property name="doc" value="doc"/>

    <path id="lib.path">
        <pathelement location="${build}" />
    </path>

    <target name="init">
        <mkdir dir="${build}"/>
    </target>
    <target name="build" depends="init">
        <javac srcdir="${src}" destdir="${build}" debug="true" includeantruntime="false">
        </javac>
    </target>

    <target name="run" depends="build">
        <java classname="scnr" fork="true" dir="." maxmemory="4096m">
            <classpath location="."/>
            <classpath refid="lib.path"/>
            <arg file="data/test.txt"/>
        </java>
    </target>

    <target name="clean">
        <delete dir="${build}"/>
    </target>
</project>
```

(3) command

- ant build

- ant run

* this build.xml already set the file name(test.txt)

2) Use Javac (directory name)

(1) java version

- openjdk version "11.0.6" 2020-01-14
- OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)
- OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode)

(2) command

- javac scnr.java
- java scnr [file name]